INFORMATICS INSTITUTE OF TECHNOLOGY

Software Development II

Coursework Report 2022

Name : Thilini Abeywickrama

UoW ID : W1830145

Student ID : 20200476

Task 01 – Source Code

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
public class Main {
    public static void main(String[]args) throws IOException {
        Scanner input = new Scanner(System.in);


        String []ServiceCenter1 = new String[13];  //Names of the passengers
are stored
        Initialise(ServiceCenter1);  //Getting ready the cabin

        File myFile = new File("CruiseRecord.txt"); //Creating a file

        int userValue=0;
        while(userValue!=999){  //This is to exit from the program. You can
use do-while loop too instead of this method
            System.out.println("To View all Cabins Available:----------------
----"+"V\n"+
                               " To View all Empty Cabins:-------------------
-------"+"E\n"+
                               " To Add a Passenger to a Cabin:--------------
-----------"+"A\n"+
                               " To Remove Passenger from a Cabin:-----------
---------"+"D\n"+
                               " To View Passengers Sorted in an alphabetical
order:-----"+"O\n"+
                               " To Store Program Data into file:------------
-------"+"S\n"+
                               " To Load Program Data from file:-------------
-------"+"L\n"+
                               " To Exit the Program:------------------------
-------"+"999 or Q");


            System.out.println("");
            System.out.println("Your selection: ");
            String user=input.next();
            switch (user){

                case "V":
                    ViewAllCabins(ServiceCenter1);
                    break;


                case "E":
                    ViewEmptyCabins(ServiceCenter1);
                    break;


                case "A":
                     AddPassenger(ServiceCenter1);
```

```
                    break;


                case "D":
                    RemovePassenger(ServiceCenter1);
                    break;


                case "O":
                    ViewPassengersSorted(ServiceCenter1);
                    break;


                case "S":
                    StoreFile(ServiceCenter1);
                    break;


                case "L":
                    LoadFile();
                    break;




                case "Q":
                    userValue = 999;
                    System.out.println("Thank you for using this cruise
management system. Have a Good day!");
                    break;

                default:
                    System.out.println("Sorry, Invalid choice. Please try
again.");
                    System.out.println("");
            }
        }
    }
    public static void Initialise(String cabins[]){
        //Getting ready all 12 cabins to store names of passengers.
        System.out.println("---Welcome to the cruise management  system!---
");
        for(int x=0;x<12;x++){
            cabins[x]="e";
            System.out.println("Cabin number "+x+" initialized!");
        }
        System.out.println("All cabins successfully initialized! Please
assign customers to each cabin.");
        System.out.println("");
    }
    public static void ViewAllCabins(String cabins[]){
        //This views all cabins  including empty cabins.
        System.out.println("---Viewing all cabins in the cruise---");
        for(int x=0;x<12;x++){
            if(cabins[x].equals("e")){
                System.out.println("Cabin number "+x+" is empty. Please
assign a customer to this Cabin.");
```

```
            }
            else{
                System.out.println("Cabin number "+x+ " is occupied by
"+cabins[x]);
            }
        }
        System.out.println("");
    }
    public static void ViewEmptyCabins(String cabins[]){
        //This views only empty cabins which can assign a new Passenger.
        System.out.println("---Viewing all empty cabins in the cruise---");
        int checkEmpty=0;  //A variable which used to check if all the cabins
are occupied or not.
        for(int x=0;x<12;x++){
            if(cabins[x].equals("e")){
                checkEmpty=1;
                System.out.println("cabin number "+x+" is empty. Please
assign a customer to this cabin.");
            }
        }
        if(checkEmpty==0){
            System.out.println("All cabins are occupied by passengers.");
        }
        System.out.println("");
    }
    public static void AddPassenger(String cabins[]) {
        //This lets you to add a new Passenger to the cabin. Also shows empty
cabins to make the adding task easy.
        System.out.println("---Viewing empty booths that you can add a new
customer---");
        for (int x = 0; x < 12; x++) {  //You can use previously declared
method(ViewEmptyBooths) here too to show empty booths
            if (cabins[x].equals("e")) {
                System.out.println("Booth number " + x + " is empty. Enter "
+ x + " to assign a customer to this booth.");
            }
        }
        System.out.println("");
        Scanner input = new Scanner(System.in);
        System.out.println("Your selection of booth number: ");
        int cabinNum = input.nextInt();

        System.out.println("Enter customer name for booth " + cabinNum + "
:");
        cabins[cabinNum] = input.next();


        System.out.println("Customer " + cabins[cabinNum] + " successfully
added to the booth.");
        System.out.println("");

    }



    public static void RemovePassenger(String cabins[]){
        //This lets you to remove a passenger. To make the task easy, it
```

```
shows occupied booths too.
        System.out.println("---Here is the current occupied cabins list---");
        ViewAllCabins(cabins);
        System.out.println("");

        Scanner input = new Scanner(System.in);
        System.out.println("Enter the cabin number that you want to free?");
        int cabinFree = input.nextInt();

        cabins[cabinFree] = "e";
        System.out.println("The passenger in the cabin number "+cabinFree+"
successfully removed from the cabins list.");
        System.out.println("");

    }
    public static void ViewPassengersSorted(String cabins[]){
        //Viewing passenger names in alphabetical order.
        System.out.println("---Viewing customer names according to the
alphabetical order---");
        int n = 6;

        String temporary;
        for(int i=0;i<n;i++){
            for (int j=i+1;j<n;j++){
                if (cabins[i].compareTo(cabins[j])>0){
                    temporary = cabins[i];
                    cabins[i] = cabins[j];
                    cabins[j] = temporary;
                }
            }
        }
        System.out.println("Sorted passengers list: ");
        for (int i = 0; i < n; i++) {
            if(cabins[i].equals("e")){
                continue;
            }
            else {
                System.out.println(cabins[i]);
            }
        }
        System.out.println("");
    }

    public static void StoreFile(String cabins[]) throws IOException {
        //Storing data of the passengers
        FileWriter myFile = new FileWriter("CruiseRecord.txt");
        for(int x=0;x<cabins.length;x++){
            if(!(cabins[x].equals("e"))){
                myFile.write("Cabin Number "+x+"\nPassenger Name:
"+cabins[x]+"\n");
                myFile.write("------------------\n");
                myFile.write("");
            }
        }
        myFile.close();
        System.out.println("Passenger details successfully updated to a
file.");
```

```
        System.out.println("");
    }

    public static void LoadFile() throws FileNotFoundException {
        //Loading cabin numbers with the names to the console
        System.out.println("--- Printing information in the file to the
console---");
        File myFile = new File("CruiseRecord.txt");
        Scanner myReader = new Scanner(myFile);
        while (myReader.hasNextLine()) {
            String data = myReader.nextLine();
            System.out.println(data);
        }
        myReader.close();
        System.out.println("");

    }
```

Task 02 – Source Code

Cruise Class

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class Cruise {

    private static final Scanner input1 = new Scanner(System.in);
    static Cabin[] cabins = new Cabin[12];

    public static void main(String[] args) {

        Passenger[] persons = new Passenger[12];
        String selected = "";

        // initialise
        setCabins(cabins, persons);

        // loop until user stops
        while (true) {
            if (!selected.equalsIgnoreCase("Q")) {
                selected = displayMenu();
                menu(selected,persons);
            } else {
                System.out.println("Thank you for visiting us.");
                break;
            }
        }
    }
```

```
    public static void setCabins(Cabin[] rooms, Passenger[] persons) {
        for (int x = 0; x < cabins.length; x++) {
            cabins[x] = new Cabin(x + 1, "empty",0);
            persons[x] = new Passenger("","","");
        }
        System.out.println("initialise ");
    }

    private static String displayMenu() {


        System.out.println("Menu :");
        System.out.println("V: ToView all cabins available");
        System.out.println("A: To Add new passenger to the cabin");
        System.out.println("E: To Display Empty cabins in the cruise");
        System.out.println("D: To Remove a Passenger from a cabin");
        System.out.println("F: To Find a Passenger in a cabin");
        System.out.println("S: To Store data into file");
        System.out.println("O: To Print to the cabins in the alphabetically
order ");
        System.out.println("L: To Load data from file");
        System.out.println("Q: To Quit");
        System.out.println(("*********************"));

        System.out.println("What do you want to do? ");
        return input1.next();
    }

    private static void menu(String selected, Passenger[] persons) {

        switch (selected.toUpperCase()) {

            case "V":
                viewCabins(cabins);
                break;

            case "A":
                addAPassenger(cabins,persons);
                break;

            case "E":
                showEmptyCabins(cabins);
                break;

            case "D":
                removeAPassenger(cabins,persons);
                break;

            case "F":
                findAPassenger(cabins,persons);
                break;

            case "S":
                saveToFile(cabins,persons);
                break;
```

```
            case "L":
                cabins = loadFile();
                break;

            case "O":
                sortByName(cabins);
                break;

            default:
                break;
        }
    }

    //Sort By Name "O"
    private static void sortByName(Cabin[] cabins) {
        String temp1;
        for (int i = 0; i < cabins.length; i++) {
            for (int j = i + 1; j < cabins.length; j++) {

                // to compare one string with other strings
                if
(cabins[i].getCustomerName().compareTo(cabins[j].getCustomerName()) > 0) {
                    // swapping
                    temp1= cabins[i].getCustomerName();
                    cabins[i].setCustomerName(cabins[j].getCustomerName());
                    cabins[j].setCustomerName(temp1);
                }
            }
        }
        printCabins(cabins);
    }

    private static void printCabins(Cabin[] cabins) {
        System.out.println("Printing the sorted Array of the names of the
passengers");
        for (int i = 0; i < cabins.length; i++) {
            System.out.println("Passenger Name: " +
cabins[i].getCustomerName());
        }
    }

    //Load from file "L"
    private static Cabin[] loadFile() {
        Cabin[] tmp1 = new Cabin[12];
        try {
            System.out.println("Enter File name to load");
            String folderName = input1.next();
            BufferedReader infile = new BufferedReader(new
FileReader(folderName));
            String l;
            System.out.println("The File contained:");
            int i = 1;
            while ((l = infile.readLine()) != null) {
                System.out.println(l);
                tmp1[i].setCustomerName(l);
                i++;
            }
```

```
            infile.close();
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        for (int x = 0; x < tmp1.length; x++) {
            if (tmp1[x].getCustomerName() == null) {
                tmp1[x].setCustomerName("empty");
            }
        }
        return tmp1;
    }

    //Write to file "S"
    private static void saveToFile(Cabin[] cabins, Passenger[] persons) {
        String folderName = "";
        while (folderName.equals("")) {
            System.out.println("Enter File Name to read");

            folderName = input1.next();
            System.out.println("File Created");
        }
        try {
            FileWriter writer = new FileWriter(folderName);
            writer.write("CabinNo, Name, NoOfGuests, FirstName, Surname,
Expenses\n");
            for (int i = 0; i < cabins.length; i++) {
                writer.write( (i + 1)+", "+cabins[i].getCustomerName()+",
"+cabins[i].getNoOfGuests()+", "+persons[i].getFirstName()+",
"+persons[i].getSurname()+", "+persons[i].getExpenses());
                writer.write("\n");
            }
            writer.close();
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }

    //Find a passenger
    private static void findAPassenger(Cabin[] cabins, Passenger[] persons) {
        System.out.println("Enter Passenger Name to find cabin: ");
        String name = input1.next();
        boolean nameAvailable = false;
        for (int x = 0; x < cabins.length; x++) {
            if (cabins[x].getCustomerName().equalsIgnoreCase(name)) {
                System.out.println(name + " is in cabin" + (x + 1));
                System.out.println("No. of passengers in cabin: "+
cabins[x].getNoOfGuests());
                System.out.println("Payment Details: ");
                System.out.println("Name: "+ persons[x].getFirstName() +" "+
persons[x].getSurname());
                System.out.println("Credit Card No.: "+
persons[x].getExpenses());
                nameAvailable = true;
            }
        }
```

```
        if (!nameAvailable) {
            System.out.println("Sorry. There is no customer named "+ name);
        }
    }

    //remove a customer from a cabin
    private static void removeAPassenger(Cabin[] cabins, Passenger[] persons)
{
        System.out.println("Enter Passenger Name to Delete: ");
        String del = input1.next();
        for (int x = 0; x < cabins.length; x++) {
            if (cabins[x].getCustomerName().equalsIgnoreCase(del)) {
                cabins[x] = new Cabin();
                persons[x] = new Passenger("","","");
                cabins[x].setCustomerName("empty");
                System.out.println("Removed " + del + " from cabin " + x +
1);
            }
        }
    }

    //Could view the empty cabins in the cruise
    private static void showEmptyCabins(Cabin[] cabins) {
        for (int x = 0; x < cabins.length; x++) {
            if (cabins[x].getCustomerName().equals("empty")) {
                System.out.println("Cabin " + (x + 1) + " is empty");
            }
        }
    }

    // Add a passenger to a cabin
    private static void addAPassenger(Cabin[] cabins, Passenger[] persons) {
        String cabinName;
        int cabinNum;
        System.out.println("Enter cabin number (1-12) or 13 to go back: ");
        cabinNum = input1.nextInt();
        if (cabinNum != 13) {
            if (cabins[cabinNum - 1].getCustomerName().equals("empty")) {
                System.out.println("Enter the name for a cabin " + cabinNum +
" :");
                cabinName = input1.next();
                cabins[cabinNum - 1].setCustomerName(cabinName);
                System.out.println("Please note that you could only include a
maximum of 3 passengers for a cabin");
                System.out.println("Enter no of passengers in a cabin " +
cabinNum + " :");
                cabins[cabinNum - 1].setNoOfGuests(input1.nextInt());
                System.out.println("Payments of the customer: ");
                System.out.println("Enter first name: ");
                persons[cabinNum - 1].setFirstName(input1.next());
                System.out.println("Enter Surname: ");
                persons[cabinNum - 1].setSurname(input1.next());
                System.out.println("Enter expenses of the passenger: ");
                persons[cabinNum - 1].setExpenses(input1.next());
            } else {
                System.out.println("The cabin " + cabinNum + " is already
occupied");
```

```
            }
        }
    }

    // View All Cabins "V"
    private static void viewCabins(Cabin[] cabins) {
        for (int x = 0; x < cabins.length; x++) { //By traversing looks
whether the cabins are occupied or not
            if (cabins[x].getCustomerName().equals("empty")) {
                System.out.println("Cabin " + (x + 1) + " is empty");
            } else {
                System.out.println("Cabin " + (x + 1) + " is occupied by " +
(cabins[x].getCustomerName()));
            }
        }

    }
}
```

Cabin class

```
public class Cabin {

    private String customerName;
    private int cabinNumber;
    private int noOfGuests;

    Cabin() {}

    public Cabin(int cabinNumber, String customerName, int noOfGuests){
        this.customerName = customerName;
        this.cabinNumber = cabinNumber;
        this.noOfGuests = noOfGuests;
    }

    public void setCustomerName(String customerName){
        this.customerName = customerName;
    }

    public String getCustomerName(){
        return customerName;
    }

    public int getCabinNumber(){
        return cabinNumber;
    }
    public void setCabinNumber(int cabinNumber){
        this.cabinNumber=cabinNumber;
    }

    public void setNoOfGuests(int noOfGuests){
        this.noOfGuests = noOfGuests;
    }

    public int getNoOfGuests(){
```

```
        return noOfGuests;
    }

}
```

Passenger Class

```
public class Passenger {

    private String firstName;
    private String surname;
    private String expenses;

    Passenger() {
    }

    public Passenger(String firstName, String surname, String expenses) {
        this.firstName = firstName;
        this.surname = surname;
        this.expenses = expenses;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getSurname() {
        return surname;
    }

    public String getExpenses() {
        return expenses;
    }

    public void setExpenses(String expenses) {
        this.expenses = expenses;
    }
}
```

Task 03 – Source Code

Did not implement

Task 04 – Testing

Test Case	Expected Result	Actual Result	Pass/Fail
(Cabins Initialized correctly) After program starts, Press 'V'	Displays "empty" for all the cabins from 0-11	Displays "empty" for all the cabins from 0-11	Pass
(Add Passenger "Thilini" to cabin 0) Select A, enter "Thilini"	Passenger added successfully with details. Press V Outputs "Thilini" I in cabin 0	Thilini is cabin 0 and correctly with details	Pass
(Add Passenger "Namal" to cabin 2) Select A, enter "Namal"	Passenger added successfully with details. Press V Outputs "Namal" I in cabin 2	Namal is in cabin 2 and correctly with details	Pass
(Check for the empty cabins in the cruise) Press E,	Cabins 2 and 3-11 should be displayed as empty	Outputs the empty cabins from 3-12	Pass
(Add Passenger "Chamal" to cabin 3. Select A, enter "Chamal"	Passenger added successfully with details. Press V Outputs "Chamal" I in cabin 3	Chamal is cabin 3 and correctly with details	Pass
Remove "Namal" from cabin 2 Select D, and enter the cabin number	Passenger removed successfully. Press V Outputs cabin 2 is empty.	Output cabin 2 is empty	Pass
(Order the passengers alphabetically) Press O	Outputs the passengers list in alphabetical order.	Displays the passengers list in alphabetical order.	Pass
Save the data of the passengers to a file.	Saves data of the customer to text file	Gives an error saying null point exception.	Fail

Press S and enter a file name you want to enter details.			
Load data from the file you entered data, Press L and enter the file name.	Loads data that was saved	Method is working but data isn't loaded because of the problem in the previous method.	Pass
Exit the program Press Q	Exits the program	Exits the program	Pass
Test Case (Task 02)	**Expected Result**	**Actual Result**	**Pass/Fail**
(Cabins Initialized correctly) After program starts, Press 'V'	Displays "empty" for all the cabins from 1-12	Displays "empty" for all the cabins from 1-12	Pass
(Add Passenger "Thilini" to cabin 1) Select A, enter "Thilini"	Passenger added successfully with details. Press V Outputs "Thilini" I in cabin 1	Thilini is cabin 1 and correctly with details	Pass
(Add Passenger "Namal" to cabin 2) Select A, enter "Namal"	Passenger added successfully with details. Press V Outputs "Namal" I in cabin 2	Namal is in cabin 2 and correctly with details	Pass
(Check for the empty cabins in the cruise) Press E,	Cabins 3-12 should be displayed as empty	Outputs the empty cabins from 3-12	Pass
(Add Passenger "Chamal" to cabin 3. Select A, enter "Chamal"	Passenger added successfully with details. Press V Outputs "Chamal" I in cabin 3	Chamal is cabin 3 and correctly with details	Pass
Remove "Namal" from cabin 2 Select D, and enter the cabin number	Passenger removed successfully. Press V Outputs cabin 2 is empty.	Output cabin 2 is empty	Pass
(Find a customer from the cabins)	Displays the details of the customer.	Displays the details of the customer.	Pass

Select F, and enter the customer's name			
(Order the passengers alphabetically) Press O	Outputs the passengers list in alphabetical order.	Displays the passengers list in alphabetical order.	Pass
Save the data of the passengers to a file. Press S and enter a file name you want to enter details.	Saves data of the customer to text file	Data saved to text file	Pass
Load data from the file you entered data, Press L and enter the file name.	Loads data that was saved	Gives a null point exception error	Fail
Implement a method to get the expenses of the passenger	Loads the expenses of the passenger	Not implemented	Fail
Exit the program Press Q	Exits the program	Exits the program	Pass

Task 04 – Testing - Discussion

Testing checks whether the actual software product matches with the expected requirements of the proposed product. It ensures that the product is bug free. Having a test plan enables you to have an idea on the aspects, testing is carried out. When creating the test plan, test coverage was also considered. Even though there are many aspects that needs to be considered to make a complete test plan such as performance testing, usability testing and etc. But here unit testing is done manually on the methods and activities that were identified in the application. In task 1, all the methods that were asked to implement are tested with necessary inputs. As you can see in the test plan all the methods are tested. The test plan all consist of the cases that weren't implemented. Same process has been carried out for the task 2 as well to ensure that all the methods that was required by the specification is up and running. Software testing is an important factor that needs to be considered when developing software development project.

Discussion on the best way to implement the project

According to myself, the best and the easiest way to implement the solution is to use the **classes version**. Classes version is developed using the OOP concepts of Java programming language. When coding using OOP concepts you need to have classes and objects. In simple form, objects are things that represent real world entities in specific domains. Classes are the blueprints that classifies the attributes and methods which should be included in objects. In the problem two classes have been used to create two objects. The 4 main concepts of Java have been effectively used here. Abstraction principles have been used in the classes "Cabin" and "Passenger" in order to represent the important factors regarding the object. It has helped in reducing memory allocation. Encapsulation principal is used here by making the variables in the classes private. When variables are made private you cannot the variables in the main class and for that you need getters and setters. It has helped with security concerns. Polymorphism concept is also used in constructor overloading. Inheritance concepts aren't used in the code. By using the class version to implement the code, it could be reused multiple times, inherit to subclasses and most importantly it is easy to maintain and modify the code. For an instance, you only have to do some modifications to expand the code. By if you go ahead with the **Array version,** it is hard to maintain and modify the code. It doesn't allow you to reuse code snippets and efficiency could be less when compared with the **classes version**.

Self-Evaluation form

Criteria	Component marks	Expected Mark
Task 1 One mark for each option (A,V,E,D,F,S,L,O) Menu works correctly	24 6	**18** **6** **24**
Student comment: All the methods have been implemented with necessary information. Storing data method gives a null point exception, so that method isn't working fully. In the loading method the file is correctly loaded but doesn't output because of the error in the previous method. Menu is implemented fully and its working correctly.		
Task 2 Cabin class correctly implemented. Passenger class correctly implemented. Expenses correctly reported.	14 10 6	**14** **8** **0** **22**

Student comment: Necessary classes have been implemented. Cabin and the passenger classes are correctly implemented with variables and methods. All the methods that were asked to implement are implemented in the cruise class which has the main method in it. Got an error in the method which allows the user to load data that was saved. The error displays a null point exception error. Tried different ways to solve it but couldn't find a way. Method to store the details of expenses isn't implemented in the code but have the given the option to store manually by having a variable in the passenger class.		
Task 3 Waiting list queue implementation "A: Add"works correctly "D: Delete"works correctly Circular queue implementation	10 3 3 4	**0**
Student comment: **Did not implement**		
Task 4 Test case coverage and reasons Writeup on which version is better and why	6 4	**5** **4** **9**
Student comment: **Test cases have been implemented in order to cover all the aspects of the code. Considered the methods that weren't implemented as well. Test cases have been developed for both the versions. Differences of the two ways of implementation have been discussed in depth. Reasoning has been provided for the choice of implementation in a comprehensive way.**		
Coding Style (Comments, indentation, style) Complete the self-evaluation form indicating what you have accomplished to ensure appropriate feedback.	7 3	**5** **3** **8**
Student comment: **Comments has been written in the code explaining the methods and variables. Proper techniques have been used in method and variable naming. Have followed the necessary ethics in coding.**		
Totals	**100**	**60-65**
Demo: At the discretion of your tutor, you may be called on to give a demo of your work to demonstrate understanding of your solutions. If you cannot explain your code and are unable to point to a reference within your code of where this code was found (i.e., in a textbook or on the internet) then significant marks will be lost for that marking component. If you do not attend a requested demo your mark will be capped at 50%.		

References

1. www.stackoverflow.com
2. www.geeksforgeeks.com
3. www.w3schools.com
4. www.tutorialspoint.com